2D VISION TECHNIQUES FOR THE HANDLING OF LIMP MATERIALS

NAGW-1333

By:

R.B. Kelley

Department of Electrical, Computer and Systems Engineering
Department of Mechanical Engineering, Aeronautical Engineering & Mechanics
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

CIRSSE Document #8

NATO Adv. Res. Workshop: Sensory Robotics for the Handling of Limp Materials, Oct. 1988

2D VISION TECHNIQUES FOR THE HANDLING OF LIMP MATERIALS

Robert B. Kelley
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590, USA

Keywords: machine vision / robot vision / binary techniques / structured illumination / grey-level techniques / handling / in-process inspection.

Abstract — For automated robotic systems to handle limp materials, sensors are needed to determine the actual properties of the material in the supply, the material itself, and at the handling destination. This paper reviews 2D vision techniques with an aim toward satisfying the sensory goals associated with the handling and in-process inspection of limp materials. The 2D vision techniques are summarized in a discussion of their applicability in the context of typical questions asked of vision guided robotic systems.

I. INTRODUCTION

For most manufacturing processes, the automated handling of parts by a machine requires the parts to be properly positioned. In the case of handling limp materials, there is a significant challenge since such materials do not satisfy the usual assumption that the part is rigid. Of course, the rigid part assumption can be handled by stiffening the limp material or modeling the behavior of the material. For sensory robotic systems to handle limp materials, sensors must be used to determine the actual properties of the material supply, the material itself, and the handling destination. A machine vision system is a particularly suitable, general-purpose sensor. It is often the sensor of choice when there is a wide variety of parameters to be measured. Here, the focus is on the 2D vision techniques which are particularly appropriate to the sensory tasks that are associated with the robotic handling of limp materials.

To motivate the 2D vision techniques which are discussed, typical questions are listed in Table 1.1. These are posed to serve as an aid in identifying the various ways 2D vision might be used to supply the desired answers. The list of questions is aimed at accomplishing typical tasks, namely locating and acquiring a piece of limp material from a supply; determining the orientation, alignment and proper placement of a piece of material; and performing limited in-process inspection of limp materials which are handled by automatic systems.

During the discussion which follows, it will be helpful to keep such questions in mind. First, some elementary 2D vision techniques are presented. These techniques are chosen with an aim toward satisfying the sensory goals which accomplish the typical tasks listed below. Then, the 2D vision techniques are summarized and discussed in the context of the typical questions posed for the handling of limp materials.

Table 1.1 Questions to motivate 2D vision technique selection

LOCATING AND ACQUIRING A PIECE OF LIMP MATERIAL FROM A SUPPLY.
0. *Where is the stack of parts?*
WHEN DEALING WITH A STACK OF PARTS:
1. *Where is the topmost part?*
2. *Where can the part be grasped?*
WHEN DEALING WITH ISOLATED PARTS:
3. *Is the part where is is expected to be?*
4. *How is the part resting?*
DETERMINING THE ORIENTATION, ALIGNMENT AND PROPER PLACEMENT OF A PIECE OF MATERIAL.
5. *What is the orientation of the part?*
6. *Is the part properly aligned?*
7. *How is the part resting?*
PERFORMING LIMITED IN-PROCESS INSPECTION OF LIMP MATERIALS.
8. *Is it the correct part?*
9. *Is it a good part?*

II. PREPROCESSING

This section begins the presentation of 2D vision techniques which can be applied to the handling of limp materials. No attempt is made to be exhaustive; rather, the purpose here is to direct attention to the variety of tools which are useful in this context.

Shading correction

Shading correction is used to transform a grey-level image into an image which approximates a uniformly illuminated one. For each pixel, the intensity measured by a camera is the product of the illumination incident on the reflecting surface, the surface reflectance, and the optical system

of the camera. Further, the pixel value obtained depends on the transfer characteristic (intensity versus output) of the camera. In general, the transfer characteristic of the camera needs to be known to compensate for non-uniform illumination. Using a constant reference light source and varying the f-stops of the camera lens, the grey-level pixel value for a range of input intensities can be measured. This series of known apertures settings may be used to estimate the transfer characteristic.

Frequently, however, the transfer characteristic is modeled as a smooth exponential function of the form

$$g = g_o(I/I_o)^\gamma, \tag{2.1}$$

where g and g_o are observed and reference grey-levels, I and I_o are observed and reference intensities. (Several contemporary solid state cameras can be adjusted to have a γ parameter value of one.) For this transfer characteristic, shading correction can be obtained by scaling observed pixel values. The individual pixel correction scale factors are found as follows. The distribution of the non-uniform illumination of the scene is measured by viewing a calibration surface with a uniform, known reflectance. Usually the maximum over all the pixel values is taken as the reference grey-level g_o. With this information, observed grey-level values can be corrected to the values which uniform illumination would have given (within the quantization error). The correction factor is given by the ratio of the reference grey-level to the calibration grey-level for that pixel. That is,

$$k_{pixel} = g_{ref}/g_{pixel} \tag{2.2}$$

$$g_{corrected} = k_{pixel}\, g_{uncorrected}\,. \tag{2.3}$$

Automatic threshold selection

Binary images may be thought of as being a special subset of grey-level images. They can be obtained from grey-level images through the application of a two-valued transformation to each pixel. Usually a simple threshold is used for the entire image to segment it into object and background pixels. The pixels with grey-levels above the threshold are assigned one value and the rest the other value. If care is taken in obtaining approximately uniform illumination or if the contrast is strong enough (for example, when back lighting is used), then a single threshold value for the entire image may be satisfactory. Clearly, it is desirable to be able to use a single threshold value rather than to use extra processing effort to compensate for non-uniform illumination. The extra effort expended to obtain high quality images is rewarded by avoiding the repeated performance of extra preprocessing steps on every image. Corrections may be made to grey-level images for non-uniform illumination. Conventionally, the image is partitioned into non-overlapping regions. The threshold is adjusted in each region to approximate the result for a uniformly illuminated scene.

Automatic threshold selection requires that the image have good contrast and the regions be separable by means of a single threshold. Automatic selection of the threshold is based on the assumption that the histogram of each region generally has both object and background pixels. Various rules may be used to select the threshold for each region. If the histogram has two well-defined peaks with a flat valley, then some valley point is chosen as the threshold. Finding an extreme value of a criterion function is an objective way to select a *good* threshold which divides the histogram into two classes, object and background. An algorithm [Duda and Hart, 1] selects the threshold T which minimizes the criterion function $J_1(T)$ given by.

$$J_1(T) = \sum_{n=0}^{T} [h(n) - m_b(T)]^2 + \sum_{n=T+1}^{255} [h(n) - m_o(T)^2] , \qquad (2.4)$$

where T is the threshold, h(n) is the histogram function, and m_b and m_o are the mean values of the background and object pixels.

More recently, the following criterion function $J_2(T)$ was suggested for threshold selection [Otsu, 2].

$$J_2(T) = \frac{n_b(T)\, n_o(T)\, [m_b(T) - m_o(T)]^2}{[n_b(T) + n_o(T)]} , \qquad (2.5)$$

where T is the threshold, $n_b(T)$ and $n_o(T)$ are the number of background and object pixels, and m_b and m_o are the mean values of the background and object pixels. This is a smooth function and usually has a single maximum. When there is more than one maximum, then each maximum is examined to determine which maxima are associated with good thresholds. Two additional criteria, valley checks, are used [Kittler and Illingsworth, 3].

$$\sum_{j=-1}^{1} h(\tau + j) < \sum_{j=-1}^{1} h[m_b(\tau) + j] \qquad (2.6)$$

and,

$$\sum_{j=-1}^{1} h(\tau + j) < \sum_{j=-1}^{1} h[m_o(\tau) + j] , \qquad (2.7)$$

where $h(\cdot)$ is the histogram function, τ is the threshold corresponding to the local maximum, and m_b and m_o are the mean values of the background and object pixels for this threshold selection. These checks compare local averages of the number of pixel values about the chosen threshold and the class means, m_b and m_o. If there are two dominant grey-levels, both valley checks will be satisfied. For some images, only one of the check conditions may be satisfied; this is a signal that the underlying two dominant grey-level assumption may not be satisfied and the maximum found may not correspond to a good threshold.

Connectivity

When images are represented as a rectangular array of pixels, the determination of which regions are connected is complicated by the restriction that the sum of the connected neighbors for an object pixel and for a background pixel equals 12. A problem arises because it is natural to consider the 8 adjacent pixels of an object pixel as its connected neighbors. This then requires that background pixels be restricted to having just 4 adjacent pixels be designated as its connected neighbors. Clearly, the roles of the two kinds of pixels can be reversed. However, 8-connected objects have intuitive connection properties. (Some prefer to work with a symmetric definition where both object and background pixels have 6 neighbors [Agin, 4]; however, the selection of the 2 adjacent, non-connected pixels leads to other interpretation anomalies such as direction sensitive diagonal connections.) The different neighborhood definitions and a sample image array are shown in Figure 2.1.

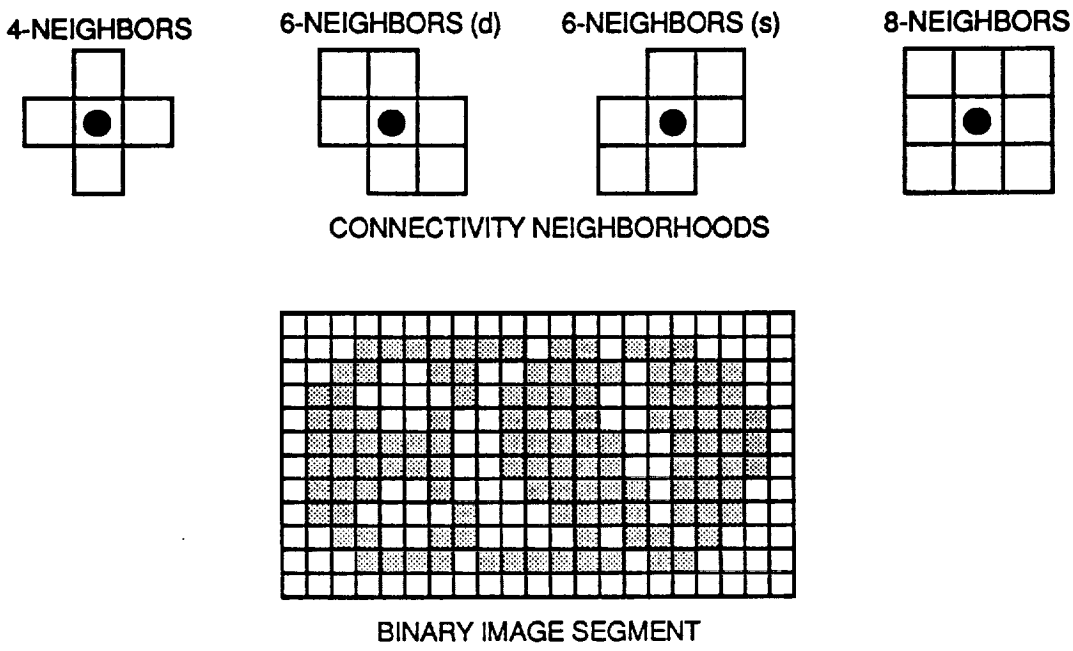

Figure 2.1. Connectivity neighborhoods and sample image array.

Erosion and dilation

Erosion and dilation are dual operations which are applied to regions in a binary image. The erosion operation removes pixels from an object region which are adjacent to a background pixel. That is, such object pixels are relabeled as background pixels. Similarly, the dilation operator relabels background pixels which are adjacent to object pixels. A comparison of

erosion-dilation operations applied in both sequences is shown in Figure 2.2. The two operators are duals since eroding an object dilates the background and vice versa. These operators have many uses. They are used for postprocessing thresholded images, for generating templates, for emphasizing or isolating irregularities, etc.

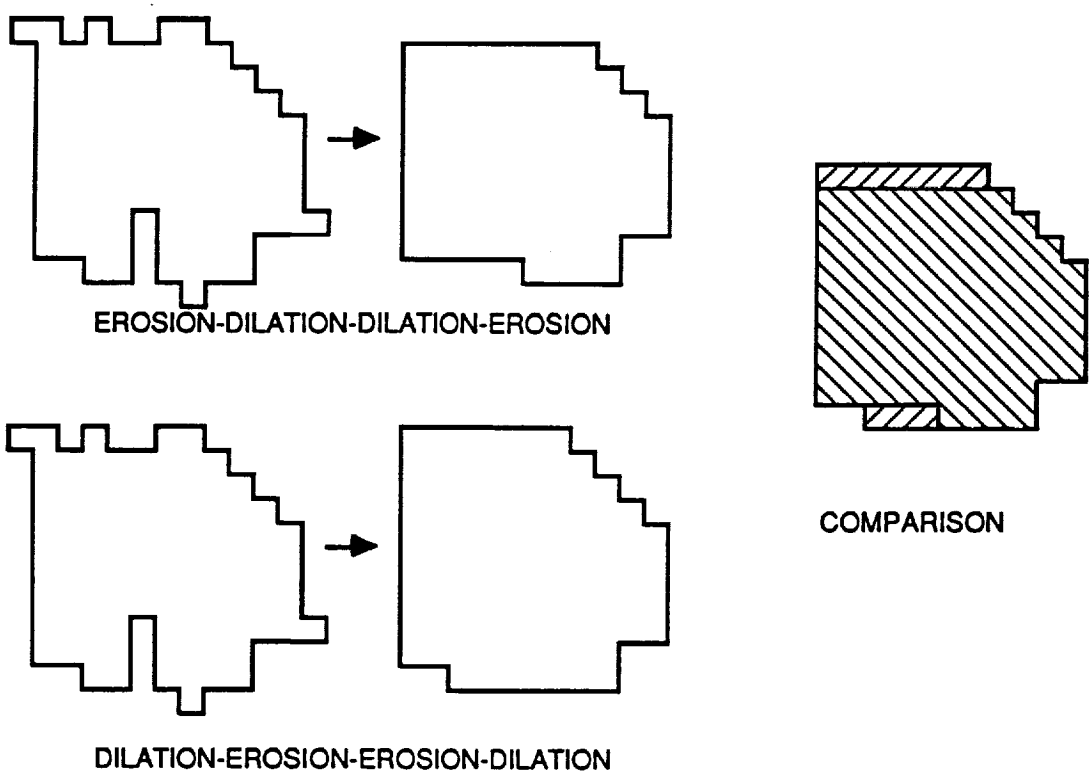

Figure 2.2. Comparison of the results two sequences of erosion and dilation operations.

Postprocessing of binary images is employed to rationalize object regions by removing artifacts produced by the thresholding operation. For example, a common defect is the presence of isolated noise pixels, extra peripheral pixels, or embedded background pixels. The erosion operator is used to remove isolated noise pixels and to shave peripheral *whisker* pixels. The dilation operator is used to fill in holes made by missed object pixels. For this purpose, a sequence of both operators is typically applied in performing the rationalization.

As defined above, clusters of noise pixels with a diameters up to 2 pixels and whiskers of 1 pixel length or 2 pixels width are removed in a single erosion operation; holes up to 2 pixels wide are filled in a single dilation operation. Thus the sequence of erosion, dilation, dilation, and erosion operations usually removes both kinds of defects and restores the object to its normal size. Alternatively, the sequence of dilation, erosion, erosion, and dilation operations can also be used. Differences between the actual performance of these two operator sequences arise when the initial operator changes the topology of the object region. Two object regions

separated by no more than 2 pixels will become one region if the dilation operation is applied. Embedded holes within 2 pixels of the boundary of the object region will become joined to the background if the erosion operation is applied; if dilation is immediately applied, the gap may be closed again.

Masks or templates can be generated from the binary image of objects in a similar manner. In fact, the rationalized object region can serve as a mask to identify embedded holes and the like. To suppress spurious responses when using such a mask, extra dilation operations can be used to make the mask oversized. In this way small whiskers and small holes will be ignored. Difference images produced by these masks are useful in emphasizing irregularities and departures from the nominal object region topological characteristics.

III. Grey-Level Image Segmentation

Grey-level images may be segmented by a variety of techniques. In this section, one geometry- and two intensity-based techniques are presented. In general, segmentation techniques have to be carefully selected to match the nature of the scene being segmented.

Structured illumination

Structured illumination is a technique which is sensitive to relative depth geometry. Thus, it can be used to segment a grey-level image by extracting silhouettes of objects as well as detecting surface irregularities on objects, such as wrinkles. The segmentation results in a binary image that is obtained by directly thresholding the grey-level image of the structured light pattern. The difference of grey-level images of the scene, with and without the structured light pattern, can be used to overcome interference from extraneous illumination. Segmentation is provided by noting the regions where there are deviations from the expected nominal structure of the projected pattern. The deviations are usually due to relative depth differences between regions of the scene. These deviations can be interpreted to obtain geometric information such as geometric edge location and orientation.

Consider a plane-of-light projector as shown in Figure 3.1. The nominal pattern for the intersection of the plane-of-light with a flat, reference surface is a straight line. Hence the nominal location of the line in the image corresponds to surfaces which are the same height as the reference. When the plane-of-light is inclined, relative surface height differences result in displacements of the line perpendicular to the direction of the nominal line pattern. The relative height difference is proportional to the displacement. (In fact, the displacement can be computed for each pixel of the line to provide a 3D slice through an object.)

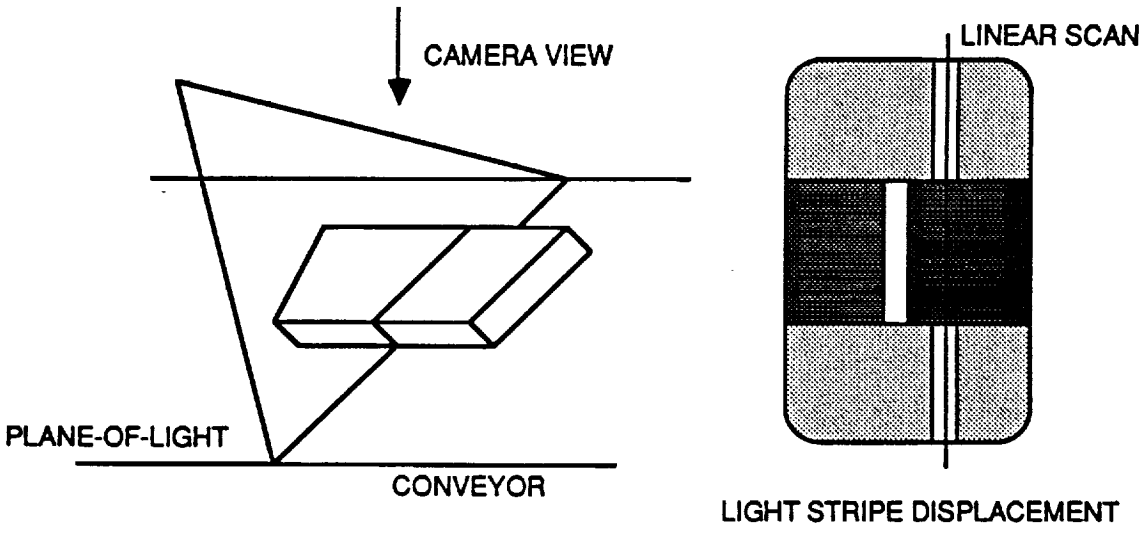

Figure 3.1. Plane-of-light silhouette extraction.

Silhouette extraction does not require any calculation. Rather the silhouette is obtained by simply noting where the nominal line pattern is broken. The number of projected plane-of-light measurements depends upon the application. A simple implementation which extracts the complete silhouette is obtained by moving the object relative to the line pattern. This can be realized by having the object on a conveyor, the projector on a track, or a robot arm carry all or part of the structured illumination and camera system. Multiple planes-of-light are a useful way to get more information from each image. A practical camera system for silhouette extraction would employ a linear array camera to provide faster scan times and simplify the silhouette extraction computation [Baird, 5].

Edge extraction

Edge extraction techniques associate large changes in grey-level values with geometric edges. Thus edge extraction is performed by means of edge detectors which are based on discrete approximations to directional derivative operators. Two orthogonal directional operators are used together as a gradient vector operator.

Popular edge detection operators include the 2x2 Roberts cross operator, and the 4-neighbor, 8-neighbor, and Sobel 3x3 operators. The Roberts cross operator effectively computes a diagonal difference which is centered at the 4-corner point, and is slightly more sensitive to higher spatial frequencies because of the size of the neighborhood [Roberts, 6]. The 4-neighbor operator computes pixel-centered orthogonal differences, has the least smoothing of the 3x3 operators. The 8-neighbor operator computes a weighted sum of orthogonal and both diagonal differences,

has intermediate level of smoothing. The Sobel operator also computes a weighted sum of orthogonal and both diagonal differences with the orthogonal difference given a relative weight of 2, has the most smoothing and the smallest gradient direction error. The directional differences, gradient magnitude and direction expressions are listed in Table 3.1.

Table 3.1 Vector edge detection operators

Operator	Directional differences	Gradient magnitude	Gradient direction
Roberts cross	$d_1 = p_{i+1,j+1} - p_{ij}$ $d_2 = p_{i,j+1} - p_{i+1,j}$	$m_{ij} = \sqrt{d_1^2 + d_2^2}$	$\theta_{ij} = \arctan(d_2/d_1) - \pi/4$
4-neighbor	$d_i = p_{i+1,j} - p_{i-1,j}$ $d_j = p_{i,j+1} - p_{i,j-1}$	$m_{ij} = \sqrt{d_i^2 + d_j^2}$	$\theta_{ij} = \arctan(d_j/d_i)$
8-neighbor	$s_i = p_{i,j-1} + p_{i,j} + p_{i,j+1}$ $s_j = p_{i-1,j} + p_{i,j} + p_{i+1,j}$ $d_i = s_{i+1,j} - s_{i-1,j}$ $d_j = s_{i,j+1} - s_{i,j-1}$	$m_{ij} = \sqrt{d_i^2 + d_j^2}$	$\theta_{ij} = \arctan(d_j/d_i)$
Sobel	$s_i = p_{i,j-1} + 2p_{i,j} + p_{i,j+1}$ $s_j = p_{i-1,j} + 2p_{i,j} + p_{i+1,j}$ $d_i = s_{i+1,j} - s_{i-1,j}$ $d_j = s_{i,j+1} - s_{i,j-1}$	$m_{ij} = \sqrt{d_i^2 + d_j^2}$	$\theta ij = \arctan(dj/di)$

Some prefer to use a scalar edge detection operator based on discrete approximations to the Laplacian as given in Table 3.2 or the Laplacian of a Gaussian smoothing filter [Horn, 7]. The argument is made that such operators improve noise suppression and edge localization properties. The tradeoff is between this performance improvement and the penalty represented by the loss of edge direction information. Commonly, an attempt is made to make up this loss by introducing external knowledge about the edge properties or some form of edge continuity assumption. In the extreme case, a scalar directional edge detector may be employed to obtain local edge direction information.

Table 3.2 Scalar edge detection operators: Laplacian approximation

Operator	Filter output
4-neighbor	$m_{ij} = 4p_{i,j} - (p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1})$
8-neighbor	$m_{ij} = 8p_{i,j} - (p_{i-1,j-1} + p_{i-1,j} + p_{i-1,j+1} + p_{i,j-1} + p_{i,j+1} + p_{i+1,j-1} + p_{i+1,j}\ p_{i+1,j+1})$

Region growing

Region growing exploits small changes in grey-level values to aggregate adjacent pixels into common regions. Under uniform illumination, such regions would be associated with (almost) planar faces on the object. If only local differences are used, there can be a wide range of pixel values clustered together in the same region. On the other hand, if distant differences are used, the segmentation which is achieved often depends on the order in which the region growing was performed. Relaxation methods are employed to minimize this latter effect.

IV. Binary Image Segmentation

Binary image segmentation requires the blobs in the scene to be identified or labeled. The highest level of description of the segmentation is given by the connected components tree. Isolated blobs can then be described in terms of their contours or by their moments (Section V).

Connected components

This analysis produces a connection tree which identifies the objects in the image, the holes entirely enclosed by each object, the objects found inside of each hole, and so forth. As a byproduct, connected component analysis usually produces parameters which are specific to each element in the connectivity tree: such as its area, centroid, moments and axes of inertia, etc. These parameters are treated in Section V. The position and orientation of any blob is easily computed from such parameters.

Contour extraction

A search technique such as contour extraction may be used to segment a binary image into distinct blobs. Contour extraction requires a starting point on the edge of the blob. For sequential computers, an easy way to locate a starting point on the blob edge is to scan along a row or column until the edge of a blob is crossed. If a reference point is known to be interior to the blob, then starting the scan for the edge from the reference point is efficient. A standard *edge following bug* is used to extract a contour description of the blob. A simple edge following bug traces the contour in a fixed direction [Ballard and Brown, 8]. At each step, it examines its neighbors in order while keeping the interior on the same side as it traces the contour. If is outside the blob, it searches for the first neighbor in the forward direction which is in the blob and conversely. This bug will fail on thin, hair-like blobs; using two bugs which trace the contour from the left side and from the right side solve this problem [Dessimoz, 9]. The

extracted contour description allows the position and orientation of the blob to be determined by techniques treated in the next section.

V. Implicit Shape Features

These features are obtained by using information which is not constrained to be local to the immediate neighbor of any particular pixel. Such features include moments, transforms, topological properties, and interfeature relationships. These features may be used to describe the shape, position and orientation, or other patterns for purposes of location or recognition.

Shape description

To describe the shape of an object, the object must be distinct from the other objects in the image; that is, the image must be segmented and the object isolated from the rest of the scene. Hence, for this discussion, it is assumed that there is only one object present. Shape may be described in terms of implicit parameters or explicit geometry-based parameters such as boundary curvature discontinuities (see Section VI).

Implicit parameter descriptions are those as given by the 2D mechanical (or statistical) moments, or by Fourier transform coefficients for the uniformly weighted pixels, for example. For single valued objects as obtained from a binary image, all the shape information is contained in its boundary relative to the background (and embedded holes). Because the interior pixels of the object dominate these parameters, some researchers treat the interior pixels as background pixels and give them zero weight. This increases the sensitivity of such parameter calculations to perturbations in the number and location of the peripheral pixels.

The lower order moments are also generally used in the preparatory steps of other computations. The zero-order moment gives the area of the object. The first-order moments give the coordinates of the center of mass, the centroid – an object reference position point. The second-order central moments are computed with respect to the centroid as the coordinate origin and provide a reference orientation direction (with an ambiguity of π).

The mechanical/statistical moments, up to the second-order, are computed as follows. Let p_{ij} be the pixels with non-zero weights. Then, the area is given by

$$m_{00} = \Sigma_i \Sigma_j p_{ij} , \tag{5.1}$$

the centroid (m_{10}, m_{01}) by

$$m_{10} = (1/m_{00}) \Sigma_i \Sigma_j i\, p_{ij} ,$$
$$m_{01} = (1/m_{00}) \Sigma_i \Sigma_j j\, p_{ij} , \tag{5.2}$$

and the central moments of inertia by

$$m'_{20} = (1/m_{00}) \Sigma_i \Sigma_j (i - m_{10})^2 p_{ij} ,$$
$$m'_{11} = (1/m_{00}) \Sigma_i \Sigma_j (i - m_{10})(j - m_{01}) p_{ij} ,$$
$$m'_{02} = (1/m_{00}) \Sigma_i \Sigma_j (j - m_{01})^2 p_{ij} . \quad (5.3)$$

The direction of the axis of the least moment of inertia is often used as a modulo-π orientation direction reference. Higher-order moments can be computed in a similar fashion. Since the third-order moments are modulo-$2\pi/3$, they can be used to disambiguate the polarity assignment for the direction of the axis of least moment of inertia.

Polar Moments

Some rotational invariant moment measures (polar moments) can be computed directly from the mechanical moments given above. Define the polar moment magnitude as

$$M_{ij} = \sqrt{C_{ij}^2 + S_{ij}^2} \quad (5.4)$$

where the C_{ij} and S_{ij} are related to the m_{ij} by the following relationships:

$C_{00} = m_{00}$	$S_{00} = 0$	$C'_{31} = m'_{30} + m'_{12}$	$S'_{31} = m'_{03} + m'_{21}$
$C_{11} = m_{10}$	$S_{11} = m_{01}$	$C'_{22} = m'_{20} - m'_{02}$	$S'_{22} = 2\, m'_{11}$
$C'_{20} = m'_{20} + m'_{20}$	$S'_{20} = 0$	$C'_{33} = m'_{30} - 3\, m'_{13}$	$S'_{00} = 3\, m'_{00} - m'_{00}$ (5.5)

Contrast invariance is provided by dividing the higher-order moments by the area, m_{00}. *Image size invariance* is provided dividing the polar moment M_{ij} by R^{2i}, where R is the radius of gyration computed as

$$R = \sqrt{(m'_{20} - m'_{02})/m_{00}} . \quad (5.6)$$

SRI-type geometry-based parameters

Originally developed at SRI in the 1970s, such features (frequently referred to as SRI features) [4] have been used to describe not only the object's shape but also its topology. Efficient algorithms were developed in the early 1970s, principally at SRI International. These algorithms produce explicit geometry-based descriptive parameters which are primarily used for feature recognition. SRI used special purpose vision capture hardware which directly output a run length representation of the binary image of the scene; this was further processed by a minicomputer. Similar vision systems were marketed in the US and Europe in the late 1970s. A *partial* list of SRI-type features is given in Table 5.1.

Table 5.1 Partial list of SRI-type features

The surface area of the object;
The length of the enclosing perimeter;
The total area of all holes;
The maximum distance measured from the centroid to perimeter pixels;
The minimum distance measured from the centroid to perimeter pixels;
The root-mean-square distance measured from the centroid to perimeter pixels;
The angle(s) between the rays from the centroid to the maximum distance perimeter pixel(s);
The angle(s) between the rays from the centroid to the minimum distance perimeter pixel(s)
The angle to the centroid of embedded holes, corner pixels, and other features, measured relative to the direction of the ray from the centroid through the maximum perimeter pixel; and
The compactness of the blob, defined as the ratio of the square of the perimeter length to the area with a minimum value of 4π for a circle.

VI. Explicit Shape Features

The shape of the perimeter of an object can be represented by a sequence of the directions of steps taken as the entire perimeter is traversed. The direction sequence is cyclic. If the starting pixel is chosen as the maximum distance pixel, for example, the sequence of relative direction changes is (ideally) unique and rotationally invariant. Because of the spatial quantization effect of array cameras, the rotational invariance is only approximate. However, if the step distances are taken into account and the Fourier Series coefficients are computed, then removing the higher-order Fourier coefficients of the relative direction sequence (that is, the sequence is smoothed by a non-causal low-pass filter) gives a slightly rounded but more rotationally insensitive representation [9].

Chain code

The chain code is a representation of the perimeter of an object in terms of the step direction as the perimeter is traced one pixel at a time. Traditionally, the directions are taken to be either the 4-neighbor or 8-neighbor directions. If the step neighborhood is extended beyond the 3x3 neighborhood, somewhat better direction angular resolution can be obtained. Alternatively, the sequence of step directions can be smoothed over a window covering several steps. In all cases, it is important to take the pixel-to-pixel distances into account or suffer severe distortions when the perimeter is rotated on the rectangular image lattice. The chain code also exhibits a discontinuity if there are 2π direction changes along the perimeter. Modulo-2π reduction must

be used to handle rotations. Differential direction chain codes are well suited for a rotation invariant shape representation.

Curve fitting

In addition, a plot of the differential directions versus the step distance around the perimeter provides an easy way to extract a constant curvature, or *concurve*, representation of the perimeter [Perkins, 10]. Thus the concurves represent the perimeter as connected segments of straight lines and circular arcs. In the differential direction vs. step distance plane, concurves form horizontal lines. Hence, the best horizontal line fit automatically finds the break points for the concurve segments.

Line fitting

Fitting a line to a set of scalar point locations is usually done according to a best-fit technique. The eigenvector with the larger eigenvalue which passes through the centroid of the cluster of points is frequently computed and gives the least square error line fit. If the points carry direction information, then the direction histogram can provide not only the direction of the best fitting line, but also identify those points which are farthest from the line. Elimination of such outliers usually improves the quality of the line fit. Hough transform techniques which are also used in scalar point straight line fitting are made greatly more efficient by the inclusion of line direction information to limit the dispersion of the Hough accumulator entries [8].

VII. Summary Discussion and Conclusion

There are a large number of techniques which might be used applied to the handling of limp materials. A brief review of some promising techniques has been given. The short list of application related questions from Section I (repeated here) give a context to the issues discussed.

0. Where is the stack of parts? This question has a trivial answer if the parts are where they are expected to be. If not, prior knowledge about the supply stack should provide a guide to the selection of appropriate 2D vision techniques. Structured light appears to be the most universal, reliable technique to answer such questions.

1. Where is the topmost part on a stack of parts? Structured light again provides the most reliable way to locate the topmost layer of fabric or leather in a stack. Otherwise, grey-level edges can be used to extract an estimate of the geometric edge locations. Binary blob extraction might also suffice.

2. *Where can the topmost part be grasped?* The answer to the question depends strongly on the gripper selected; the part itself can constrain the available holdsites. Surface attachment grippers, such as needle point devices, require a site which is virtually flat and of sufficient size. Structured light and binary blob extraction are the most promising approaches. Clamping grippers need to locate at least one edge of the topmost part. Structured light is indicated again, with grey-level edge detection a reasonable alternative.

3. *Is the isolated part where is is expected to be?* This question requires the part position to be be checked. A binary image blob centroid location is sufficient. Similar computations can be performed on contour or edge data.

4. *How is the isolated part resting?* This requires the part orientation to be determined. Again blob moments, especially polar moments, are usually sufficient. Explicit geometric features such as corner and hole locations can provide robust orientation estimates.

Position and orientation corrections. The displacement between the actual and nominal feature centers or centroids can be used, together with robot eye-to-hand calibration, to update the nominal location commands in the robot control program. Any rotational correction which may be needed is obtained by applying the camera horizontal and vertical scale factors to the components of the actual and nominal reference direction vectors. Rotational corrections can also be computed by projecting the image coordinates onto a robot coordinate system calibration plane. (Note that angular measurements are sensitive to image coordinate scale factors, but are independent of the origin of the coordinate system employed.)

5. *What is the orientation of the part?* Often it is prudent to verify the orientation of a part while it is in the grasp of the gripper. Parts sometimes suffer from minor slippage when initially grasped. Question 4 techniques can be applied to this case. The presence of the gripper in the image needed to be accommodated.

6. *Is the part properly aligned?* This is also a verification step to check the part in the gripper and is approached in a manner similar to that in question 3.

7. *How is the part resting?* This is a verification of the quality of the placement of the part at its target location. It includes position, orientation and alignment with other parts. If questions 5 and 6 have been answered, then only the relative alignment needs to be verified. Structured light is the indicated approach. Grey-level edge detection is a reasonable alternative to detect misaligned edges. Prior knowledge needs to be exploited to simplify the interpretation of the possibly cluttered scene image.

8. Is it the correct part? There is always a need to verify the correctness of the part being handled. This is simplest when the part is isolated or being held in the gripper. If the question is asked in response to an alignment failure, then the task is more difficult. However, the extraction of the most salient features can simplify getting the answer.

9. *Is it a good part?* This is a variation of question 8 which focuses on the quality of the correct part or on its placement. For example, surface irregularities such as wrinkles can be detected by using structured light patterns with an area array camera. Departures from the nominal 2D pattern provide the means to locate surface smoothness irregularities. By isolating the departures from the nominal pattern, the direction of the wrinkle and, hence, the direction to apply the smoothing action can be determined. Small wrinkles could be ignored, if appropriate, by noting the magnitude of the pattern deviation.

Table 7.1. Suggested techniques versus typical question

	Salient features	Blob/polar moments	Contour/edge data	Binary blob data	Grey-level edge data	Structured lighting
0. Where is the stack of parts?	.	.	.	.	.	X
1. Where is the topmost part on the stack?	.	.	.	X	X	X
2. Where can the topmost part be grasped?						
surface gripper	.	.	.	X	.	X
clamping gripper	.	.	.	.	X	X
3. Is the isolated part where it is expected to be?	.	.	.	X	X	X
4. How is the isolated part resting?	X	X	.	.	.	X
5. What is the orientation of the held part	X*	X*	.	.	.	.
6. Is the held part properly alignmented?	.	.	X*	X*	.	.
7. How is the part resting?	.	.	.	.	X	X
8. Is it the correct part?	X	X	.	.	.	.
9. Is the part good?	X	X	.	X	X	X

* The presence of the gripper in the scene must be accommodated.

Conclusion

Of the many 2D vision techniques which are applicable to the vision guided handling of limp materials, the use of structured light patterns, binary blob analysis, and simple edge extraction techniques seem to be able to provide answers to the typical question encountered in applications. The suggested techniques are listed for each question in Table 7.1. Other, more sophisticated techniques may be required to handle special cases which fall outside the simplified scenario treated in this paper.

REFERENCES

1. Duda, R. and P. Hart: Pattern classification and scene analysis: New York: Wiley-Interscience 1973

2. Otsu, N.: A threshold selection method from gray-level histograms: IEEE Trans. Systems, Man, Cybernetics, **SMC-9**, 62-65 (1979)

3. Kittler, J. and J. Illingsworth: On the threshold selection using clustering criteria: IEEE Trans. Systems, Man, Cybernetics, **SMC-15**, 652-655 (1985)

4. Agin, G.: Vision systems. In: Handbook of industrial robotics, S. Nof (ed.), pp. 231-261: New York: Wiley 1986

5. Baird, M.: Image segmentation techniques for locating automotive parts on a conveyor belt: 5th Intl. Joint Conf. on Artificial Intelligence, Cambridge, MA: 694-695 (1977)

6. Roberts, L.: Machine perception of three-dimensional solids. In: Optical and electro-optical information processing, J. Tippett et al. (eds.): Cambridge, MA: MIT Press 1965: 159-197

7. Horn, B.: Robot Vision: New York: Wiley 1986

8. Ballard, D. and C. Brown: Computer vision: New York: Prentice-Hall 1982

9. Dessimoz, J-D.: Traitment des contours en reconnaissance de forms visuelles (in French): Switzerland: Ecole Polytechnique Federale de Lausanne, Ph.D thesis 1980

10. Perkins, W.: A model-based vision system for industrial parts: IEEE Trans. Computers, **C-27**, 126-143 (1978)